soniKKs
ULTRASONICS TECHNOLOGY

# User Manual
# Display and Firmware
# Cleaning Generator with rotary encoder

soniKKs GmbH
ULTRASONICS TECHNOLOGY

# Table of contents


Parameter security ..... 3

Menu items and operation ..... 3

Status LEDs ..... 4

Navigation chart ..... 5

Screens overview ..... 8

1. Power ..... 8
2. Display ..... 8
3. Info ..... 9
4. Setup ..... 9
   - 4.1. User settings ..... 10
   - 4.2. Advanced settings ..... 12
   - 4.3. Reseller settings ..... 14

Operation over Modbus/TCP ..... 19

1. Basic network information ..... 19
   - 1.1. IP Configuration Mode ..... 19
   - 1.2. IP address ..... 19
   - 1.3. Netmask ..... 20
2. Modbus/TCP protocol overview ..... 20
   - 2.1. Modbus/TCP pattern ..... 20
   - 2.2. Data types in Modbus ..... 21
   - 2.3. Modbus memory map and functions ..... 22
   - 2.4. Modbus exceptions ..... 22
3. Operating a soniKKs Cleaning Generator over Modbus/TCP ..... 23
   - 3.1. Configuring the generator's network settings ..... 23
   - 3.2. Specifications and adjustments ..... 24
   - 3.3. soniKKs cleaning generator's Modbus map ..... 24
   - 3.4. Practical examples ..... 28
4. Errors ..... 30
5. Technical Support ..... 30

Troubleshooting ..... 35

Maintenance and care ..... 36

Warranty ..... 36

Imprint ..... 37

# Parameter security

The unit features a unique three-level security concept based on numeric passwords.

- User settings: Password-Level 1
- Intermediate settings: Password-Level 2
- Reseller settings: Password-Level 3

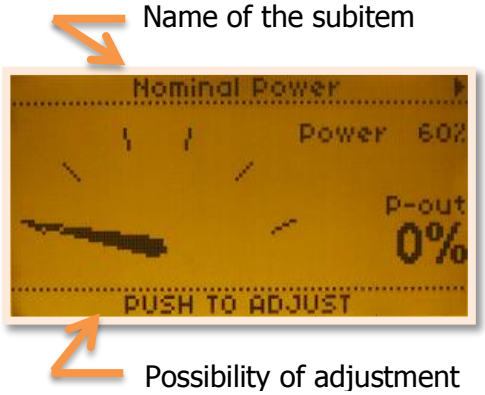
Please ask your retailer or your manufacturer for the passwords.

# Menu items and operation

The generator is operated by a rotary encoder, which can be turned as well as pushed. This makes the operation very easy and uncomplicated.

The menu is structured in several subitems (see navigation chart). You can change between the items by turning the rotary encoder.

In the first line of the display you see the **name of the subitem**, in the last line you can see whether the screen is only a display or if **adjustments can be done** (see also color-code of the navigation chart). If you can make settings, there is written "PUSH TO ADJUST" or "PUSH TO RESET".

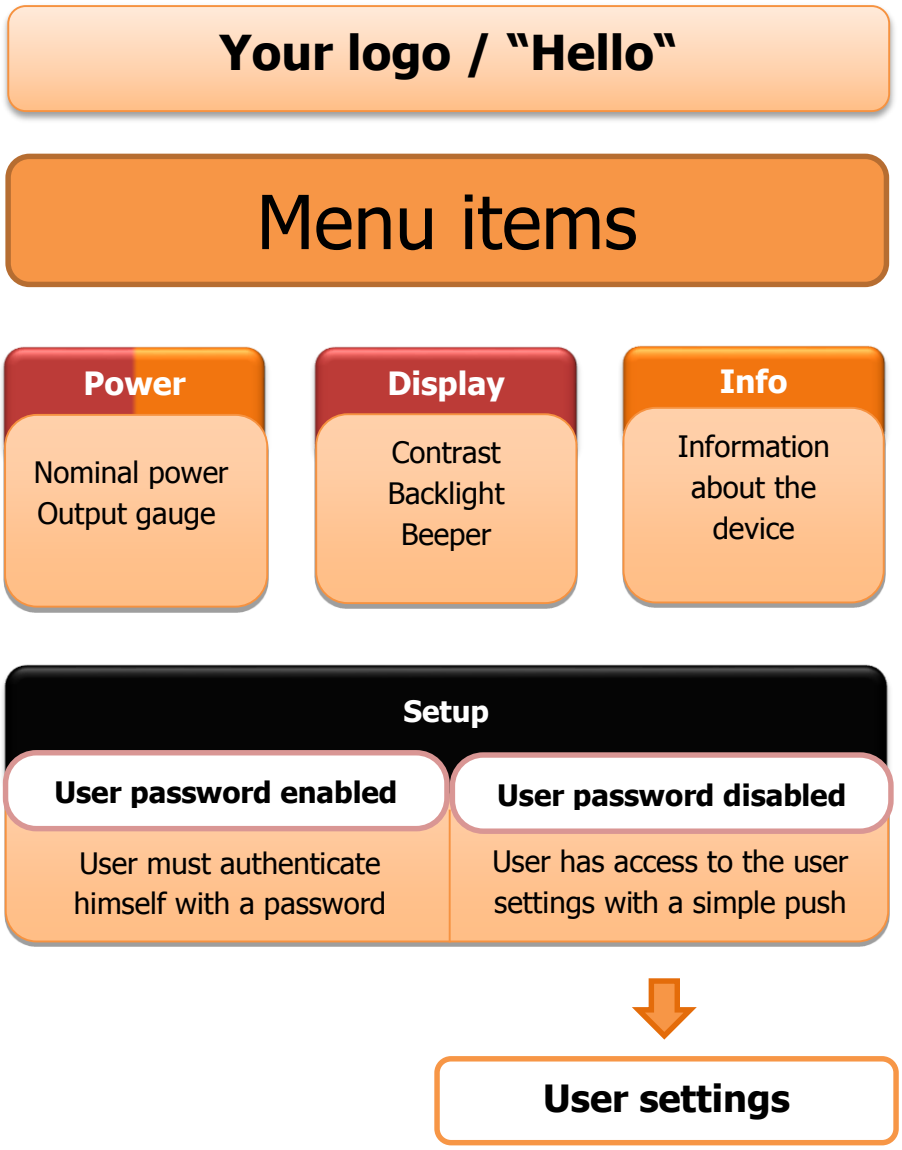



To activate the adjustment mode, push the rotary encoder as denoted in the screen. The adjustment of the values or the selection of the options is done by turning the rotary encoder. Pushing the rotary encoder another time either brings you to the next parameter's adjustment or closes the adjustment mode if there is only one adjustable parameter in the screen.

# Status LEDs

| | |
|---|---|
| **Power – LED** | Indicates that the generator is powered. |
| **Sonic - LED** | Indicates that ultrasonic power is active. |
| **Error – LED** | ➔ON: there is a malfunction of the generator.<br>➔Blinking: power level not reached during the last run |
| **Temp. – LED** | ➔ON: the generator has overheated and the electronic overtemperature fuse has shut off the generator.<br>➔Blinking: the generator shut down during the last cleaning process due to overtemperature. If you start sonic again the LED disappears. |
| **Mode – LED** | ON: the generator is set to stop after a predetermined condition has been reached. This condition can be a delay or an energy threshold. |
| **LED - gauge "Output"** | Shows the effective ultrasonic power in steps of 10% of the nominal power. |

# Navigation chart



# Settings

## User settings

| Degas | Nominal Source | Date/Time |
|---|---|---|
| On timer<br>Off timer | Ext voltage<br>Front / Bus | Date and time |

| Sensor info | Language | Advanced Setup |
|---|---|---|
| Ext. amp.<br>Gene. Temp.<br>Ext. temp. | English, German, French, Spanish, Polish | To the intermediate or reseller settings |

| Exit |
|---|
| Back to menu items |

## Intermediate settings

| Frequencies | Passwords | Factory settings |
|---|---|---|
| Adjust start/stop frequency | (de)-activate user password<br>Modify LVL 1&2 passwords | Restore factory settings |

| Exit |
|---|
| Back to menu items |

Transition

Display

Adjustment is possible

## Reseller Settings

| Testbutton | Remote start mode | Error relay polarity |
|---|---|---|
| Off<br>Push<br>Latch<br>Trigger | Latch<br>Static<br>Trigger | Normally open<br>Normally closed |

| Sonic relay polarity | Error log | Passwords |
|---|---|---|
| Normally open<br>Normally closed | List of the errors that occured | (de)-activate user password<br>Modify LVL 1&2 |

| Network Settings | Greetings | Parameter locking |
|---|---|---|
| Config. Mode<br>IP address<br>Netmask<br>MB Pipeline<br>MB Exceptions | Hello/Logo | Lock parameter adjustment in certain screens |

| Exit |
|---|
| Back to menu items |

Transition

Display

Adjustment is possible

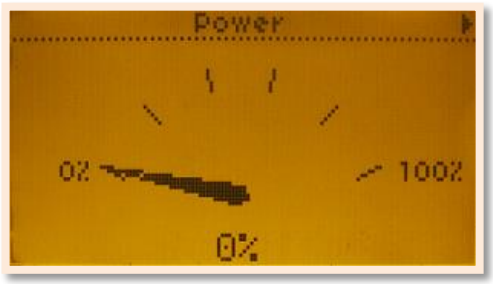
# Screens overview

## 1. Power

In this screen, you will find a gauge which indicates the actual power level emitted by the generator.

The layout of this screen depends on how you choose to control the nominal power level (see "Nominal source" screen).

If the nominal power is controlled by an external voltage, the screen features the gauge along with its digital value. No adjustment mode is available for this screen in this configuration.




If the nominal power is set via the encoder or a communication bus, the screen also displays the nominal power level.

By pushing the encoder, you can enter adjustment mode to modify this value in real-time.

Push again to quit the adjustment mode.

## 2. Display

Here you can tune display-related parameters such as the contrast and the backlight. You can also activate/de-activate the beeper.

Push the rotary encoder to enter adjustment mode.

Select the desired contrast level by turning the encoder and push it to confirm the change.

Repeat the same operation for the backlight. As you will notice, you can choose between the three options below:

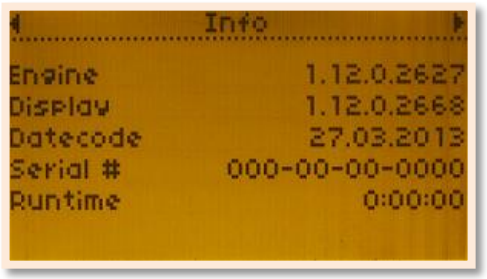
- ✓ On: the backlight is switched on permanently
- ✓ Auto: the backlight switches automatically off after a while if the generator is not used
- ✓ Dim: the backlight is dimmed to a low level

Push again to set the displayed option and go on to the beeper adjustment. Choose between "On" and "Off".

When ready, push one last time to apply the change and quit the adjustment mode.

## 3. Info

Here you can see some information about the device:




- ✓ Engine: Firmware revision of the generator
- ✓ Display: Firmware revision of the display
- ✓ Datecode: date of production
- ✓ Serial#: Serial-number of the device
- ✓ Runtime: (total) operating time (h:mm:ss)

## 4. Setup

To change the settings you need numeric passwords. Please ask your reseller or manufacturer if you do not know them.

In the navigation chart, you can see which settings require which password-level.

- ➢ User settings: Password-Level 1
- ➢ Intermediate settings: Password-Level 2
- ➢ Reseller settings: Password-Level 3

If the user password is disabled (see "Passwords" screen), no password is required in this screen. Pushing the encoder will directly bring you to the user settings. The higher security levels are reached from the "Advanced setup" screen.

If it is enabled, you will be asked a password. If you want to reach the user settings screens for instance, you will have to enter the level 1 password.

Pushing the rotary encoder will lead you to the screen on the right, with the first digit displayed. Turn the encoder until you reach the desired number. When ready, push the encoder to edit the next digit.

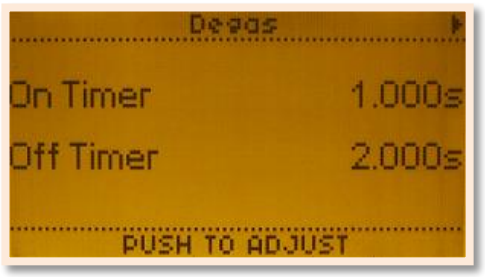
As you can see, the passwords used are 5-digit long. Repeat the same operation for all of them.

Please note that a wrong combination will lead you to an error message.

## 4.1. User settings

- **DEGAS**

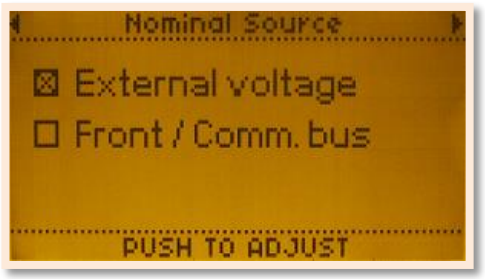
This screen contains the ON and OFF timers, which cause – when used together – the generator to emit power cyclically. The emitting and idle periods correspond respectively to the specified ON and OFF timer values.




When the testbutton or the remote is in "Trigger" mode, the ON timer has a slightly different purpose (see "Testbutton" and "Remote start mode" screens). The OFF timer becomes unused.

- **NOMINAL SOURCE**

The nominal power can be controlled by an external voltage or tuned via the user interface: the encoder and the communication buses.




Push the rotary encoder to activate the adjustment-mode. The currently set option is now highlighted. By turning the encoder, you can switch between the two options. When the desired option is highlighted, push the encoder again. Herewith you confirm the setting and exit the adjustment-mode.

- ✓ External voltage: the amplitude can no longer be adjusted with the rotary encoder.
  Instead it is set by an external voltage 5-10 V (standard) or rather 0-10 V (user specific adjustment), that corresponds to 50-100% output power.

- ✓ Front/Comm. Bus: the nominal power can directly be adjusted via the rotary

encoder or the different communication buses.

### ➢ DATE AND TIME

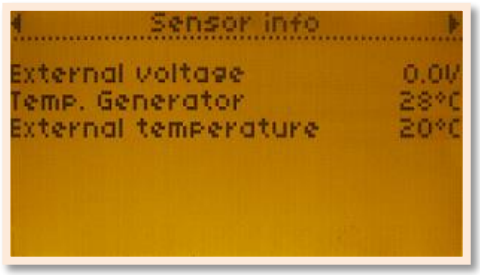
In this screen, you can read and modify the date and time of your generator's real time clock.

Using correct information and keeping the real time clock updated is important.

For instance, this date and time will be fed into the error log, which might be used by soniKKs for the maintenance of the device.

### ➢ SENSOR INFO

This screen displays various information: the external voltage that can be used to control the nominal power (see "Nominal source" screen), the internal temperature of the generator and the external temperature.




### ➢ LANGUAGE

Here you can choose the language in which the display contents are displayed.

Our generators are currently configurable to four different languages: English, German, French and Spanish.

Proceed as usual to activate the adjustment mode and change the settings.

ATTENTION: For your own safety, do not choose a language you do not understand. The language changes immediately by confirming your choice and the menu appears in the chosen language, which could be foreign to you.

- **ADVANCED SETUP**

This screen allows to reach the intermediate and reseller settings. You will especially use it when the user password is disabled (see "Passwords" screen). Indeed, in this case, these settings are not reachable from the "Setup" screen anymore.

The procedure to enter a password is the same as for the Setup screen.

- **EXIT**

Pushing the rotary encoder makes you leave the settings and brings you back to the "Power" screen.

## 4.2. Advanced settings

- **FREQUENCY**

The start and stop frequencies are adjusted by the manufacturer and should not be modified unless the generator has a problem.

In this case, please discuss any modification with the manufacturer before.

- **PASSWORDS**

The level 1 and level 2 settings are by default protected by user-customizable passwords. These passwords can be read and modified in this screen.

Also, under particular circumstances, the user might want to be granted free access to the level 1 settings. This can be achieved by adjusting the level 1 access parameter.

You can choose between the two options below:

- ✓ Secure: in the "Setup" screen, a password is required to enter any of three settings (including level 1).
- ✓ Open: in the "Setup" screen, no password is asked. Pushing the encoder will directly lead you to the level 1 settings. The higher level settings are reached from the "Advanced setup" screen.

Please note that in any of the configurations above, the "Advanced setup" screen can lead you to the level 2 and level 3 settings, given the correct password is entered.

Proceed as usual to activate the adjustment mode and change the settings.

When the level access parameter is set to "open", the third line will be hidden.

### ➢ FACTORY SETTINGS

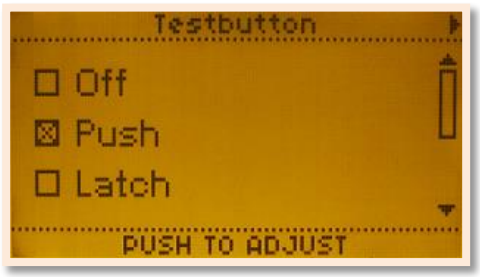
A confirmation dialog will appear if you push the encoder. By choosing yes, the factory settings will be restored.

### ➢ EXIT

Pushing the rotary encoder makes you leave the settings and brings you back to the "Power" screen.

## 4.3. Reseller settings

### ➢ TESTBUTTON




In this display you can adjust how the testbutton ("Test") works. Four options are available: "Off", "Push", "Latch" and "Trigger".

Proceed as usual to activate the adjustment mode and change the settings.

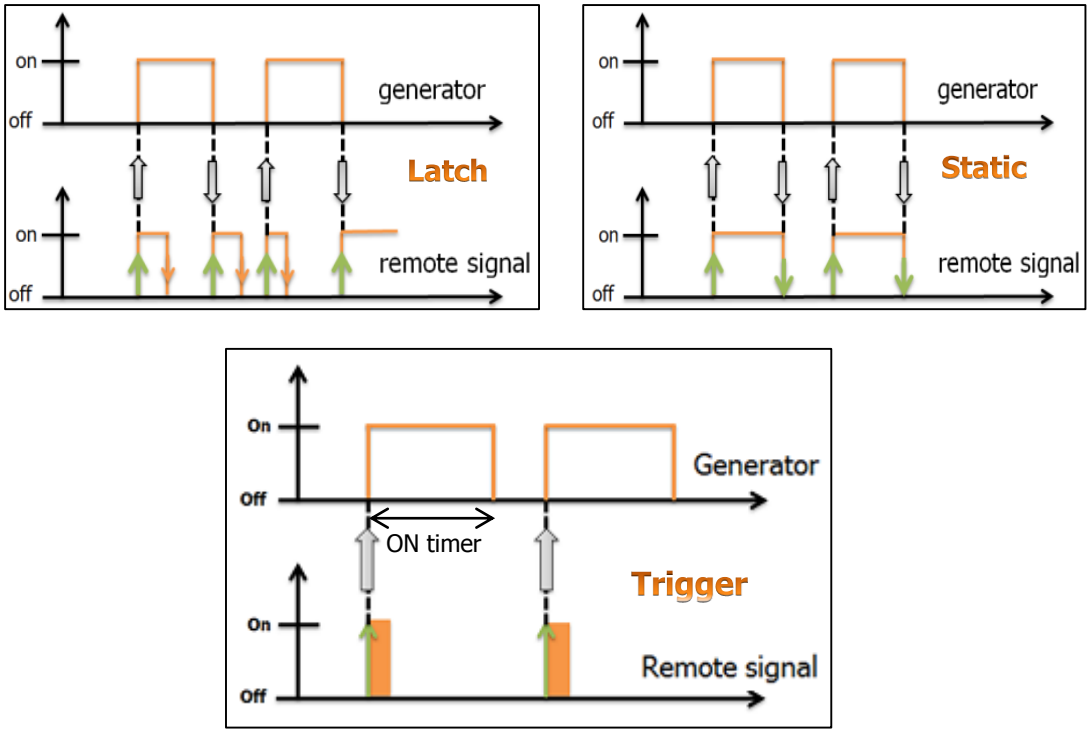
- ✓ Off: the testbutton is unused
- ✓ Push: the generator is ON as long as the testbutton is pushed
- ✓ Latch: by pushing the testbutton, you switch the generator on, pushing again switches the generator off
- ✓ Trigger: this option is only recommended when the ON timer is set (see "Degas" screen). A short (trigger)-impulse starts the generator. The device automatically switches off when the predetermined time is reached.
  - ➔ see draft in item-description "Remote start mode"

### ➢ REMOTE START MODE

Here you can adjust how the remote signal works. You can choose between the options "Latch", "Static" and "Trigger".

Proceed as usual to activate the adjustment-mode and change the settings.

- ✓ Latch: activating the remote switches the generator on.
  activating the remote again switches the generator off.
  - ➔ Storage function
- ✓ Static: the generator directly switches on or off if there is a remote signal.
  - ➔ No storage function
- ✓ Trigger: this option is only recommended when the ON timer is set (see "Degas" screen). A short (trigger)-impulse starts the generator. The device automatically switches off when the predetermined time is reached.




## ➢ ERROR RELAY POLARITY

In this screen, you can adjust the behavior of the internal error relay. The current configuration is indicated by a ticked check box.

The relay can either behave as a normally open or a normally closed contactor.

"Normally open" means that the relay will be closed upon failure of the generator. Conversely, the relay will be opened upon failure if the relay polarity is set to "normally closed".

## ➢ SONIC RELAY POLARITY

In this screen, you can adjust the behavior of the internal sonic relay. The current configuration is indicated by a ticked check box.

The relay can either behave as a normally open or a normally closed contactor.

"Normally open" means that the relay will be closed

upon start of the generator. Conversely, the relay will be opened upon stop if the relay polarity is set to "normally closed".

## ➢ ERROR LOG

Here you can have an overview of the errors which have occurred on your device.

Each entry is referenced with the date and time the error happened and a small description of the error. In some cases, this description can be replaced by an error code.

Only the last one hundred errors are kept in memory.

Please note that there is no adjustable parameter in this screen. Pushing the encoder only allows you to scroll through the log entries.

## ➢ PASSWORDS

The level 1 and level 2 settings are by default protected by user-customizable passwords. These passwords can be read and modified in this screen.

Also, under particular circumstances, the user might want to be granted free access to the level 1 settings. This can be achieved by adjusting the level 1 access parameter.

You can choose between the two options below:

- ✓ Secure: in the "Setup" screen, a password is required to enter any of three settings (including level 1).
- ✓ Open: in the "Setup" screen, no password is asked. Pushing the encoder will directly lead you to the level 1 settings. The higher level settings are reached from the "Advanced setup" screen.

Please note that in any of the configurations above, the "Advanced setup" screen can lead you to the level 2 and level 3 settings, given the correct password is entered.

Proceed as usual to activate the adjustment mode and change the settings.

When the "LVL 1 access" parameter is set to "open", the third line will be hidden.

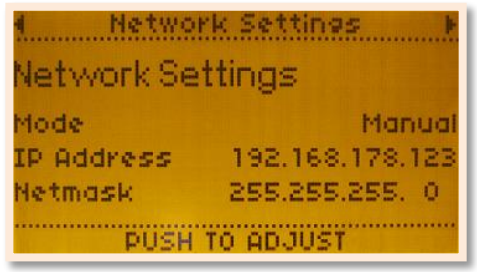
## NETWORK SETTINGS

This screen deals with the generator's network parameters. It only concerns cleaning generators which feature an embedded server.

If your generator does not have any, the message on the right will be displayed.

Otherwise, the current configuration of your device will be displayed and adjustable.

Please refer to the soniKKs generator's Modbus addendum for complete information about this feature.




Depending on the configuration, three or less parameters are displayed :

- ✓ Mode: the IP configuration mode (Manual, AutoIP, DHCP, BOOTP…). Choosing a manual configuration mode will bring you to set an IP address and a netmask yourself. Any other mode is an automatic mode, which means that these parameters are provided automatically.

If the mode is set to manual, the two parameters below are adjustable:

- ✓ IP address: the IP address used by the device on the network. It can only be set when the mode is set to Manual. In automatic mode, this line will not be displayed.

- ✓ Netmask: the netmask used by the device on the network. It can only be set when the mode is set to Manual. In automatic mode, this line will not be displayed.

Once you have set the parameters above, two additional Modbus/TCP-related settings appear:

- ✓ Modbus Pipeline: the pipeline behavior of your Modbus/TCP connection can be adjusted by turning the encoder. When set, the generator will queue the polls it receives and answer

all of them in turn. When disabled, the generator only answers the last received poll and discards all stale polls. This feature is enabled by default.

✓ Modbus Exceptions: traditional Modbus uses silence to signal some errors (e.g. unconfigured slave address, timeout, CRC error…). Setting this parameter will cause the generator to issue error messages instead, specifying the cause of the error.

### ➢ GREETINGS

This screen allows you to choose what will be displayed on startup. You can either select to display the stored logo or a simple "Hello" message.

### ➢ PARAMETER LOCKING

The parameter locking enables the supervisor to restrict access to the following parameters: the Nominal Power and the ON/OFF Timers.

In order to lock them, you have to lock the corresponding screen (e.g. "Degas" for the timers).

When a screen is locked, the bottom bar then shows "LOCKED" instead of "PUSH TO ADJUST". Trying to adjust parameters while the "parameter locking" is active will lead to an audio warning and of course the access will be refused.

### ➢ EXIT

Pushing the rotary encoder makes you leave the settings and brings you back to the "Power" screen.

# Operation over Modbus/TCP

The soniKKs cleaning generators can be equipped with an embedded server that enables them to be operated over TCP/IP using the Modbus/TCP protocol. For these devices, the user can configure the network settings as explained in this chapter.

## 1. Basic network information

### 1.1. IP Configuration Mode

The IP address of the device can be set manually (in which case it is called a "static" address) or automatically by an auto-configuration server, commonly a DHCP server (it is then called a "dynamic" address).

In manual addressing mode, the user sets the IP address himself, and can additionally set a netmask (see explanations below). This "static" configuration will remain the same as long as the user does not change it again.

The automatic addressing mode includes different modes that correspond to different auto-configuration protocols:

- ✓ **AutoIP:** the server automatically generates its own IP in the 169.254.x.x range with a Class B subnet.
- ✓ **Dynamic Host Configuration Protocol (DHCP):** a DHCP configuration server will automatically provide the server with an IP address taken from its own pool of available addresses.
- ✓ **Bootstrap Protocol (BOOTP):** a BOOTP configuration server will automatically provide the server with an IP address taken from its own pool of available addresses.

NB: The user can enable either one of those modes or several modes at the same time.

### 1.2. IP address

The IP address is an identifier that allows a device to connect to an IP network. Each device present on the network has a unique IP address in order to be reached easily.

The Modbus/TCP servers integrated in soniKKs generators use an IPv4 address that consists of a 32-bit word that is usually given in the "dotted decimal" notation: four decimal numbers ranging from 0 to 255 separated by dots (ex: 192.168.178.123).

The IP address contains two pieces of information that are the **network number** and the **host number**, which is the actual identifier of the device on the network.

The size of these numbers, which means the number of bits dedicated respectively to the network and the host ID, is defined by the first three bits of the address as follows.

| Network Class | Leading bits | Size of network number bit field | Size of host number bit field |
|---|---|---|---|
| A | 000 | 8 | 24 |
| B | 010 | 16 | 16 |
| C | 110 | 24 | 8 |

### 1.3. Netmask

In IPv4, the netmask, also called "subnet mask", is a 32-bit word used to divide a network into subnetworks. As for the IP address, it is commonly given in the dotted decimal notation.

This mask is used to indicate which part of the IP address corresponds to the network address, and thus which part corresponds to the host identifier. The "ones" delimit the bit field of the network whereas the "zeros" indicate the host's bit field.

## 2. Modbus/TCP protocol overview

NB: For more detailed information about the Modbus protocol, you may want to refer to the official documentation available at http://www.modbus.org/

Modbus is a "Master-Slave" protocol used for supervisory control and data acquisition (SCADA), in which the master (usually a PLC) can obtain and modify parameters contained in the slave's memory via "Read/ Write" commands. Only the master can send commands to the slave which executes it and systematically sends back a response (even for Write commands).

### 2.1. Modbus/TCP pattern

A typical Modbus/TCP frame is formatted as shown below. It is composed of a 7-byte header that contains indications about the message, followed by the actual data to be transmitted.

| | Transaction ID | Protocole | Length | Slave ID | Function | Data |
|---|---|---|---|---|---|---|
| Example: | 00 $58_h$ | 00 $00_h$ | 00 $06_h$ | $01_h$ | $04_h$ | 0B BE 00 $06_h$ |

The different packets have the following significations:

| | Length | Description |
|---|---|---|
| **Transact. ID** | 2 bytes | On a network, a master can send many requests in a short time and the answers to these requests may come in another order. This identifier allows the master to link each response it receives to one command it has sent. Thus, each request has a unique identifier that the response also contains. |
| **Protocole** | 2 bytes | These 2 bytes indicate which protocol is used. 00 $00_h$ corresponds to Modbus/TCP. |
| **Length** | 2 bytes | The number of bytes contained in the rest of the message (slave ID, Function and Data). |
| **Slave ID** | 1 byte | The device address on the network. Each slave present on the network has its own ID number (from 1 to 247), allowing the master to indicate the recipient of the command (i.e. to which device it is addressed). Note that messages sent to slave 0 are processed by all the slaves. |
| **Function** | 1 byte | The code for the function that the slave must perform. The Modbus specification defines a set of functions (see below) and assigns a code to each of them. |
| **Data** | N bytes | Additional data accompanying the function code such as the address of the starting register, the number of registers concerned and the data to be written (for Write commands). |

NB: the values are in hexadecimal (marked "h"), that is to say that 2 digits represent 1 byte.

## 2.2. Data types in Modbus

Modbus differentiates 4 data-types based on length and accessibility criteria.

- Coils: The discrete inputs are one-bit values that can be modified as well as read (e.g. on/off command, enable/disable a mode…).
- Discrete Inputs: The discrete inputs are one-bit values that cannot be modified by the master. They usually are status and thus can only be read (e.g. run status, error status…).
- Input Registers: The input registers are read-only values that can occupy one or more registers depending on their length. They are usually used for sensor measurements (current temperature, current power…).
- Holding Registers: The holding registers are read/write values that can also occupy several registers. They are usally used for configurable parameters such as nominal values, thresholds…

## 2.3. Modbus memory map and functions

The parameters are stored in the slave's memory in 16-bit registers. The Modbus protocol defines a memory map that separates the 4 data-types depicted above. Therefore, the memory is subdivided in 4 blocks of registers which contain our parameters. In the array below, you can see that each of these blocks can be accessed using its appropriate functions. For instance, the coils (registers 1 to 9999) can be read thanks to function 01 only.

| Function | Code | | Register Addresses |
|---|---|---|---|
| **Read Coils** | $01_d$ | $01_h$ | 1 - 9999 |
| **Write Single Coil** | $05_d$ | $05_h$ | |
| **Write Multiple Coils** | $15_d$ | $0F_h$ | |
| **Read Discrete Inputs** | $02_d$ | $02_h$ | 10001 - 19999 |
| **Read Input Registers** | $04_d$ | $04_h$ | 30001 - 39999 |
| **Read Holding Registers** | $03_d$ | $03_h$ | 40001 - 49999 |
| **Write Single Holding Register** | $06_d$ | $06_h$ | |
| **Write Multiple Holding Registers** | $16_d$ | $10_h$ | |

In the messages, however, the master always specifies an address between 0 and $9998_d$ ($270E_h$). When processing the message, the slave will then add an offset corresponding to the block implied by the function code: 1 for the coils, 10001 for the discrete inputs, 30001 for the input registers and 40001 for the holding registers.

## 2.4. Modbus exceptions

The Modbus protocol also features some error recognition cases, called exceptions. When a command is sent, there are 4 possible outcomes:

- It does not reach the slave → the master gets no response.
- It reaches the slave with some glitches → parity or CRC error → the master gets no response.
- It reaches the slave with no transmission error and is successfully processed by the slave → the master gets a valid response.
- It reaches the slave with no transmission error but the command cannot be processed → the master gets an exception response indicating the cause of the failure.

The exception response has the following pattern:

| | Transaction ID | Protocole | Length | Slave ID | Function | Data |
|---|---|---|---|---|---|---|
| Master: | 00 58$_h$ | 00 00$_h$ | 00 06$_h$ | 01$_h$ | 03$_h$ | 00 00 00 0A$_h$ |
| Slave: | 00 58$_h$ | 00 00$_h$ | 00 03$_h$ | 01$_h$ | 83$_h$ | 02$_h$ |

In this example the master asks slave number 1 to read the content of the holding registers area (03$_h$) starting at address 40001$_d$ (00 00$_h$) and including 10$_d$ registers (00 0A$_h$).

The exception is recognizable thanks to the function field. It is signaled by the slave by setting the first bit of the function code to 1. The nature of the exception is then indicated by the data field: 02$_h$ corresponds to an Illegal Data Address for instance.

Please refer to the Modbus official documentation for the description of all possible exceptions.

# 3. Operating a soniKKs Cleaning Generator over Modbus/TCP

## 3.1. Configuring the generator's network settings

To access your device over the network, here is how to proceed:

- Ask your administrator for this information:
  - ✓ An available IP address
  - ✓ The netmask
- Start your generator.
- Go to the Network Settings screen in the level-3 settings.
- Set the configuration mode to "Manual" and enter your IP address and netmask.
- Now configure the Modbus/TCP parameters (pipeline, exceptions). First, try with the default settings.
- Confirm the changes.
- Power-cycle your generator in order for the changes to be applied.

- Connect your Modbus/TCP master and wait until the left LED of the Ethernet port is steady green.

## 3.2. Specifications and adjustments

Modbus/TCP connections to the generator should be established using TCP port #502.

A **500ms connection timeout** makes the connection restart if no poll has been performed in this time frame. Thus, please make sure that the **poll rate respects this timeout**, otherwise the connection will restart indefinitely (see Technical Support on page 35).

The efficiency of the Modbus/TCP protocol is well-known. However, it requires an accurate adjustment of the master and slave settings. Therefore, you will have to give great attention to your master's tuning in order to optimize the link between your master and the generator. If your knowledge about Modbus/TCP is limited, you can refer to the paragraph called "Recommended poll rates" of the Technical Support.

Regarding the slave (the generator), two parameters are adjustable:

- The **pipeline** option: in pipeline mode, the slave processes every command present in its TCP stack, one after the other. When the option is deactivated, the slave only responds to the newest command and ignores all other commands of the stack. This way, it allows the master to retry old requests without risking building up a stale queue of waiting requests.
  The pipeline option is enabled by default. You should disable it only if you are having problems.

- The **exceptions** option: Traditional serial Modbus uses silence to signal some errors. However, Modbus/TCP defines error codes to make the errors explicit. The soniKKs generator can return two exceptions, depending on the gravity of the error: 0x0A and 0x0B. 0x0A indicates a "hard" error where a retry is not likely to succeed. Conversely, 0x0B stands for a "soft" error where a retry may succeed.

## 3.3. soniKKs cleaning generator's Modbus map

The parameters below are accessible at the specified addresses (see also section "Modbus memory map and functions"). To simplify the notations, only the lower part of the address is shown in the tables (e.g. 44018 → 4018). **Given the nature of the Modbus addressing scheme, the address sent in the message should be:**

**(register address – 1)**

### 3.3.1. Coils (Read/Write)

| Register address | Number of registers | Type | Name | Description |
|---|---|---|---|---|
| **1001** | 1 | Bit | Start/Stop command | 0: Start the generator<br>1: Stop the generator |
| **1002** | 1 | Bit | *reserved* | *reserved* |
| **1011** | 1 | Bit | *reserved* | *reserved* |

### 1.1.1. Discrete Inputs (Read only)

| Register address | Number of registers | Type | Name | Description |
|---|---|---|---|---|
| **2001** | 1 | Bit | HF Status | 0: No HF detected on output<br>1: HF detected on output |
| **2002** | 1 | Bit | ON/OFF Status | 0: The generator is not emitting power<br>1: The generator is emitting power |
| **2003** | 1 | Bit | Error Status | 0: Status OK<br>1: Status Error |
| **2004** | 1 | Bit | Start Command Status | 0: No start command pending<br>1: Start command pending |
| **2005** | 1 | Bit | Overtemp Status | 0: Temperature OK<br>1: Temperature exceeded |
| **2006** | 1 | Bit | Run Status | 0: Generator is idle<br>1: Generator is running(1) |
| **2007** | 1 | Bit | reserved | reserved |

(1) Please note that the Run Status bit can be "1" even if the generator is not currently emitting power. Example: if an OFF Timer is set, the ON/OFF status will be "1" during "ON periods" and "0" during "OFF periods" whereas Run Status will be "1" during the whole run.

### 3.3.2. Input Registers (Read only)

| Register address | Number of registers | Type | Name | Description |
|---|---|---|---|---|
| **3001** | 1 | Short | Status code | Current error code |
| **3002** | 1 | Short | *reserved* | *reserved* |
| **3003** | 1 | Short | Actual Power | Actual power level (‰) |
| **3004** | 1 | Short | *reserved* | *reserved* |
| **3005** | 1 | Short | Generator Temperature | Internal temperature (°C) |
| **3006** | 1 | Short | *reserved* | *reserved* |
| **3007** | 2 | Long | Actual Frequency | Working frequency (Hz) |
| **3009** | 2 | Long | *reserved* | *reserved* |
| **3011** | 2 | Long | Actual Timer | Elapsed time during the current run (ms) |
| **3013** | 1 | Short | *reserved* | *reserved* |
| **3014** | 2 | Long | *reserved* | *reserved* |
| **3101** | 1 | Short | *reserved* | *reserved* |
| **3102** | 1 | Short | Total Power | Maximum total output power (W) |
| **3110** | 2 | Long | *reserved* | *reserved* |
| **3112** | 2 | Long | *reserved* | *reserved* |

### 1.1.2. Holding registers (Read/Write)

| Register address | Number of registers | Type | Name | Description |
|---|---|---|---|---|
| **4001** | 1 | Short | *reserved* | *reserved* |
| **4002** | 1 | Short | *reserved* | *reserved* |
| **4004** | 1 | Short | *reserved* | *reserved* |
| **4005** | 1 | Short | *reserved* | *reserved* |
| **4006** | 2 | Long | Start Frequency (2) | Start frequency (Hz) |
| **4008** | 2 | Long | Stop Frequency (2) | Stop frequency (Hz) |
| **4010** | 2 | Long | *reserved* | *reserved* |
| **4012** | 2 | Long | ON Timer | ON time (ms) |
| **4014** | 2 | Long | OFF Timer | OFF time (ms) |
| **4016** | 2 | Long | *reserved* | *reserved* |
| **4018** | 2 | Long | *reserved* | *reserved* |
| **4020** | 2 | Long | *reserved* | *reserved* |
| **4022** | 1 | Short | Nominal Power | Nominal power level (%) |
| **4023** | 1 | Short | *reserved* | *reserved* |
| **4024** | 2 | Long | *reserved* | *reserved* |
| **4026** | 1 | Short | *reserved* | *reserved* |

(2) Please do not change these parameters unless you understand the implications.

## 3.4. Practical examples

Here are some actual examples that will help you start operating the soniKKs Cleaning Generator over Modbus/TCP.

### 3.4.1. Switching the generator ON and OFF

This can be done by setting/resetting the value contained in register 1001, using the "Write single coil" function ($05_h$). The actual address to be sent is then $1000_d$ (1001 – 1) which gives 03 $E8_h$.

To set and reset a coil, the values to send are respectively FF $00_h$ and 00 $00_h$.

Therefore, the message should look as follows:

| | Transaction ID | Protocole | Length | Slave ID | Function | Data |
|---|---|---|---|---|---|---|
| Master: | 00 $AA_h$ | 00 $00_h$ | 00 $06_h$ | $02_h$ | $05_h$ | 03 E8 FF $00_h$ |
| Slave: | 00 $AA_h$ | 00 $00_h$ | 00 $06_h$ | $02_h$ | $05_h$ | 03 E8 FF $00_h$ |

The slave returns the same frame (as for every write commands).

### 3.4.2. Adjusting the output power level

The output power level is located in register $4022_d$, and can be accessed using the "Read Holding Register" and "Write Holding Register" functions (respectively $03_h$ and $06_h$).

To read it, the master should send the following packet:

| | Transaction ID | Protocole | Length | Slave ID | Function | Data |
|---|---|---|---|---|---|---|
| Master: | 00 $AB_h$ | 00 $00_h$ | 00 $06_h$ | $05_h$ | $03_h$ | 0F B5 00 $01_h$ |
| Slave: | 00 $AB_h$ | 00 $00_h$ | 00 $05_h$ | $05_h$ | $03_h$ | 02 00 $53_h$ |

The master requests the slave number 5 to read in the holding register area (function $03_h$) starting at address 0F $B5_h$ ($4000_d$) 00 $01_h$ ($1_d$) register.

The data returned by the slave contains the length of the holding register in bytes ($02_h$) followed by the content of the register: 00 $53_h$ → $83_d$% of output power.

To tune it to 57% for example ($39_h$), the packet should be the following:

| | Transaction ID | Protocole | Length | Slave ID | Function | Data |
|---|---|---|---|---|---|---|
| Master: | 00 AC$_h$ | 00 00$_h$ | 00 06$_h$ | 05$_h$ | 06$_h$ | 0F B5 00 39$_h$ |
| Slave: | 00 AC$_h$ | 00 00$_h$ | 00 06$_h$ | 05$_h$ | 06$_h$ | 0F B5 00 39$_h$ |

The slave returns the same packet as it is a write command.

NB: The output power level can only be set between 50% and 100%.

### 3.4.3. Setting timers

The ON and OFF timers can be adjusted via the registers 4012$_d$ and 4014$_d$.

Instead of setting them with two separate "Write Holding Register" commands, you might want to adjust both with a single command.

In that case you can use the "Write Multiple Registers" function (10$_h$) as follows:

| | Trans. ID | Protocole | Length | Slave ID | Func. | Data |
|---|---|---|---|---|---|---|
| Master: | 00 AD$_h$ | 00 00$_h$ | 00 0F$_h$ | 01$_h$ | 10$_h$ | 0F AB 00 04 08 00 00 03 E8 00 00 07 D0$_h$ |
| Slave: | 00 AD$_h$ | 00 00$_h$ | 00 06$_h$ | 01$_h$ | 10$_h$ | 0F AB 00 04$_h$ |

When writing multiple coils or registers, the length of the data to be written has to be specified. Here, the master asks slave 1 (01$_h$) to write to multiple holding registers (10$_h$), to the area starting at address 4012$_d$ (0F AB$_h$) and involving 4 registers (00 04$_h$), the 8 following bytes (08$_h$): 00 00 03 E8 00 00 07 D0$_h$.

As the ON Timer includes registers 4012$_d$ and 4013$_d$, it will receive the first 4 bytes (each register is 2-byte wide). The ON Timer is then set to 00 00 03 E8$_h$ → 1000$_d$ms. The last 4 bytes will go to register 4014$_d$ and 4015$_d$, which means that the OFF Timer will be 00 00 07 D0$_h$ → 2000$_d$ms.

Note that this method is **only applicable to consecutive registers**.

The response from the device is the same frame without the written values.

# 4. Errors

In case an error occurs during the configuration (on startup), an error message will be displayed **in the Network Settings screen** and the server configuration will remain unchanged.

You may want to power-cycle your generator again. If the problem persists, please contact us.

# 5. Technical Support

Using the soniKKs generator with Modbus/TCP is normally easy. However, if a problem occurs, it is difficult to troubleshoot without an in-depth knowledge of Modbus and the system dynamics of polling.

## ❖ Some general guidelines for troubleshooting:

- Start polling slowly and increase speed gradually.
- Keep in mind that in many situations, the host application declares "No response" when in fact, the device did respond, and the application did not understand the response.

## ❖ Recommended poll rates

We recommend using the following values to configure your Modbus/TCP master, assuming that your network load and latency times are within the range of a few msec.

- response timeout: 275ms
- poll rate for fast-updating process values: 300ms
- poll rate for slow-updating process values (e.g. temperature, error code): 1000ms
- commands etc.: on demand

If you find stable results with these values when running the link over a longer period of time (2h+) you can try to lower the poll rate while keeping the response timeout untouched to allow for congestions to be gracefully resolved.

See below for a description of timeout factors to take into account.

## ❖ How fast can I poll?

It is important to understand that the link to the generator's processor is realized through a gateway from Modbus/TCP to a low-speed serial link. Therefore, you cannot expect to poll any faster than you could by a direct serial link. In fact, since you are adding a number of queuing systems between your application and device, you may even lose a bit of performance. For example, some download tests showed that remote download by Modbus/TCP bridged to Modbus/RTU ran about 20 percent slower than direct download by Modbus/RTU.

Above all, remember that the serial speed (or baud rate) consumes the largest amount of time. Suppose you issue a Modbus poll for 125 registers. This requires a 255-byte response, which at 115.2kbps (the speed of the serial link) requires over 22msec just to physically shift across the wire.

The overall time it takes to poll is the combined sum of these delays:

a. Delay for master/client to recognize need for poll.
b. Delay to issue and get the poll onto the Ethernet.
c. Delay for the poll to cross Ethernet and arrive error-free at the device (may include retries and contention).
d. Delay for Modbus/TCP gateway to process and queue Modbus/RTU poll.
e. Physical delay to shift poll bit-by-bit across the serial link.
f. Delay in the device to recognize, process, and start reply.
g. Physical delay to shift response out bit-by-bit across the serial link.
h. Delay for Modbus/TCP gateway to process and queue Modbus/TCP response.
i. Delay for the response to cross Ethernet and arrive error-free at the master/client (may include retries and contention).

Delay **a** is defined by your OPC or DDE driver. For example, a driver that runs only once each 55 msec (using the old DOS timer slice) can have a variable delay here of between 0 to 110 msec.

Delays **c** and **i** are defined by the complexity, load and link speed of your TCP/IP network. For example, if you are going through radio or satellite links, these delays routinely amount to 1000 msec (1 sec) or more per poll and another 1000 msec for a response.

Delays **e** and **g** are defined by the baud rate and the amount of data you want to transfer per request.

Delay **f** is defined by the device. The soniKKs implementation starts the reply within less than 5 msec.

Delays **d**, **e**, and **h** are defined by the load on the Modbus/TCP gateway.

### ❖ I cannot get a slave response

The soniKKs generator only expects and processes genuine Modbus/TCP from the network. Some applications just pack Modbus/RTU raw in TCP – this is not supported! Make sure that you choose "Modbus/TCP" as the protocol in your Modbus master application instead of "Modbus/RTU over TCP".

### ❖ What to do if polls fail

The soniKKs generator core probably cannot accept a new poll as fast as you are sending it over TCP.

TCP/IP is a full-duplex channel, and since you can have up to 8 active sockets, it is very easy to have a new request already waiting as your last poll's response is being returned. The only

solution to this is to slow down your Modbus/TCP masters so that they never poll before the last poll's response has been seen. This manually creates the time delay between polls your device expects.

- My soniKKs generator runs fine for about 10 minutes and then my applications report it off-line.
- My soniKKs generator runs fine until it goes off-line, then it polls only intermittently.
- Sometimes my soniKKs generator returns the wrong data.
- After a while, the soniKKs generator seems to take longer and longer to answer. After a few hours, it takes 10 minutes or more for systems changes to propagate up to the master/client.

All these relate to the same issue: a mismatch in queuing behavior and expectation by the master/client to the new realities of Ethernet (it is not the generator behaving poorly). Resetting the generator fixes the problem (flushes the bloated TCP queues full of stale requests).

The core problem is that the master/client is using the old RS232/RS485 serial assumption that no answer means poll was lost. However, in the case of the Modbus/TCP gateway in between, it could also mean the gateway has not had time to answer because it is being overworked.

Also remember that TCP is reliable – the gateway receives all polls sent without error. The result is that the master/client retries, which makes it harder for the gateway to catch up.

Here is the scenario that is causing the problem:

1. Master sends out MB/TCP Poll #A with a timeout of 1000 msec.
2. Gateway receives the poll, but the serial link is busy so it waits - possibly another MB/TCP master is being serviced or timeouts waiting on off-line stations are creating a backlog of new requests.
3. After approximately 850 msec, the serial link is now free and the gateway forwards the MB/RTU request to the controller.
4. The gateway receives the response from the generator, and since the timeout on the gateway and master are not inherently synchronized, the gateway sends the MB/TCP response into the TCP socket.
5. In the best of times, it may take 5-10 msec for this response to actually go down the gateway's TCP stack, across the wire, and up the master's TCP stack. If a WAN or satellite is involved, it could take 750 msec or longer.
6. Meanwhile, before the master receives the Response #A, it gives up and makes the Modbus/RTU assumption that the request must have been lost. The master sends out a new MB/TCP Poll #B.
7. A few msec later, there is a response that looks like a good Response #B, but actually is Response #A. If the master does not use a sequence number (which many do not) and has forgotten about pending poll #A, it wrongly assumes this is response #B (possibly with catastrophic results if Poll #B was the same size but different register range). Here is the source of the problem "Generator returns the wrong data."
8. The master is idle and has no outstanding polls. Yet the gateway has received Poll #B by TCP/IP. It sends this out to Modbus/RTU slave and gets an answer, which means it's working normally and as expected.

9. The gateway then returns Response #B to the master (if the socket is still open) and there it sits in its TCP/IP buffer. The master is not expecting more responses, so it neither receives nor purges the "extra" response.
10. Master sends Poll #C and magically finds "a response" waiting as soon as it looks in the receive buffer - yet this is stale Response #B received before poll #C was even issued. If the master does not implement Modbus/TCP sequence numbers, then it accepts the response #B as satisfying poll #C.

Imagine if the master is putting out 300 polls per minute (5 polls per second), but the gateway can only process on average 290 of those per minute and some carry over. After 10 minutes, you may have up to 100 "stale" responses waiting in your master's TCP buffer. This makes it appear as though there is now a 20-second "lag" in data reaching the master. Here is the source of your "data taking longer and longer to propagate to Master/Client" problem.

However, if the master does implement Modbus/TCP sequence numbers, then the stale responses are rejected. If the master is smart enough to resynchronize itself (Response #B does not kill poll #C, but master waits more), then this resynchronization will manifest itself as the slave going off-line and back on-line intermittently. If the master is not smart enough to resynchronize, once this out-of-sync behavior occurs, your slave goes permanently off-line.

As you can see, this Modbus/TCP master is out of sync and the only cure may be to either restart the master or power cycle the soniKKs generator. Both actions close the socket and purge the backlogged messages.

The use of a gateway brings out this shortcoming in master/client Modbus/TCP designs, but even a pure MB/TCP-to-MB/TCP network would suffer from this problem if the poll cycle approached the average response time. Any Modbus/TCP network going through WAN will discover this.

Ideally the Modbus/TCP master application used should implement the sequence number and gracefully handle receipt of stale responses with unexpected sequence numbers. Unfortunately, the Modbus/TCP specification says that this sequence number is optional and can be used by a master to match responses to requests; however it can usually be just left as zero. The Modbus/TCP slave just echoes this back in the response. Anyway, most Modbus/TCP OPC servers today do not implement the sequence number.

So what is your solution if your Modbus/TCP master is first generation?

- Slow down your poll rate.
- You have to consider the worst-case response time – assume all polls timeout.
- If you are only polling a single slave (or poll one slave at a time), then you can try to disable the "MB Pipeline" option in the server configuration (see page 22). This will either help or make things hopelessly worse. If your OPC server or host application relies on pipelining to send more than one outstanding poll at once, then disabling the pipeline will essentially stop all data communication (in which case, you can just turn the pipeline back on !).

The ideal solution is for your Modbus/TCP master to not only support the Sequence Number, but also support the receipt of the 0x0A and 0x0B extended Modbus/TCP exception response. Then

the master never needs to do retries – for each poll, it will receive either a value Modbus/TCP response or a Modbus/TCP exception that the slave is unreachable or timed out. This prevents the master from sending more polls than the gateway can process and building the TCP buffer queue up in the first place. You can enable the "use extended exception responses" in the server configuration as well (see page 22).

# Troubleshooting

| Error message | Possible reason | Repair |
|---|---|---|
| | ➢ Transducer broken<br>➢ Short circuit at RF-cable<br>➢ Error on the generator<br>➢ Start frequency too close to resonance frequency | ➢ Replace transducer<br>➢ Replace RF-cable<br>➢ Contact soniKKs<br>➢ Select higher value for start frequency |
| | ➢ Transducer broken<br>➢ Error on the generator | ➢ Replace transducer<br>➢ Contact soniKKs |
| | ➢ Transducer not connected or broken<br>➢ Cable broken | ➢ Connect or replace transducer<br>➢ Replace cable |
| | ➢ Fan broken<br>➢ Error on the generator | ➢ Check fan<br>➢ Check if air suctioning works<br>➢ Contact soniKKs |

# Maintenance and care

There is no need for special maintenance! Please remove any dust and dirt with a damp woven fabric. Do not use any chemicals to clean the device. Keep the cooling slots clear all time.

# Warranty

The period and extent of the warranty is part of the commercial terms and conditions. Special agreements are part of the confirmation of the order.

The warranty does not cover malfunctions, injuries and damages that result from misuse, unauthorized modifications, and external causes such as acts of nature.

**Warranty exclusion applies in the following cases:**

- Damage due to wrong handling or tampering
- Damage due to wrong application
- Damage due to shock, dirt or moisture
- Damage due to operation by non-qualified staff
- Damage due to nonobservance of safety regulations or accident prevention regulations
- Damage due to modifications to the operating manual

# Imprint

## Purpose and use

The operating manual explains the handling and operation of the display and the firmware in connection with standard accessories for use in laboratories and the industry. Please read especially the safety instructions carefully and observe them all.

The operating manual should always be at hand to help you solve any questions or problems that may arise.

## All rights reserved

This manual has been prepared with all due care, nevertheless faults and omissions cannot be fully precluded.

soniKKs GmbH reserves the right to make changes to the technical data and specifications during the course of further development of the product without given prior notice.

## Address

soniKKs ® Ultrasonics Technology GmbH
Neuenbürgerstraße 72
75335 Dobel
Germany

Phone: + 49 (0) 7083 – 92 48 360
Fax: + 49 (0) 7083 – 92 48 370
Mail: info@sonikks.de

www.sonikks.de
www.sonikks.com

## Editing

first edition: June 2013
second edition: September 2013